CTN Test Report

91-011

AFTB-ID-90-012

THE SEMANTIC ANALYZER

April 30, 1991

Prepared for
Air Force Logistics Command
Air Force CALS Test Bed (LMSC/SBC)
Wright-Patterson AFB, OH 45433-5000

CTN Test Report
91-011

AFTB-ID-91-012

THE SEMANTIC ANALYZER

April 30, 1991

Prepared By
SOFTWARE EXOTERICA CORPORATION/CENTECH for
Air Force CALS Test Bed
Wright-Patterson AFB, OH 45433

CTN Test Report
91-011 **AFTB-ID-91-012**

THE SEMANTIC ANALYZER

April 30, 1991

Prepared By
SOFTWARE EXOTERICA CORPORATION/CENTECH for
Air Force CALS Test Bed
Wright-Patterson AFB, OH 45433

AFTB Contact
Gary Lammers
(513) 257-3085

CTN Contact
Mel Lammers
(513) 257-8882

Prepared for

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

THE SEMANTIC ANALYZER

1. Purpose of the Semantic Analyzer

Once a document is marked up using an SGML-defined markup language, an SGML parser can validate the document against a specified Document Type Definition (DTD). This process ensures that the document's tags conform to the structure imposed by the DTD.

An SGML parser is a powerful tool for one class of markup problems, namely SGML syntactical errors. However, in the markup of CALS documents, many errors are still made, and go unchallenged, for lack of a similar tool for detecting them.

Within the confines of its structural check, an SGML parser is powerless to examine the semantics of a document -- in other words, the parser cannot assess the relevance of a given piece of text inside a particular element. Besides SGML-related markup rules, each DTD entails certain application-specific syntactic requirements -- in the case of CALS, requirements that go beyond those specified in the MIL-M-28001 DTD.

For instance, no SGML requirements are violated when a CALS paragraph is left empty, but this condition is still syntactically wrong for a CALS application. This syntactic problem is a symptom of an underlying semantic problem -- an empty paragraph is meaningless.

The Semantic Analyzer enables more powerful document analysis, combining the power of an SGML parser with the comprehensive pattern-matching capabilities of Exoterica's XGML Translator. A specification language is provided for the Semantic Analyzer, in order to allow development of customized analysis programs for specialized SGML applications.

2. Common Problems Discovered by the Semantic Analyzer

The following examples illustrate the various capabilities of the Semantic Analyzer, with reference to the MIL-M-28001 tagging specification.

2.1 Empty Elements

The Semantic Analyzer will report instances of any element which contains absolutely no content, unless the element is declared to be "EMPTY" in the DTD. For instance, the following excerpt from a CALS document contains a problem.

```
<PUBNO SCILEVEL="0"><USER SCILEVEL="0"></USER>
<DOCNO SCILEVEL="0"></DOCNO></PUBNO>
```

In the example above, the "user" and "docno" elements are empty, contrary to the intent of the document designer. Text should appear inside these elements.

The "pubno" element, on the other hand, is not considered to be empty, since it contains two sub-elements. This is not a hard and fast rule; consider the following example.

```
<PARA0><FTNOTE ID="1478">This procedure is explained at length in related technical documentation.</FTNOTE></PARA0>
```

Here, "para0" is still considered empty, because the Semantic Analyzer specification for MIL-M-28001 deems the "ftnote" element to be insufficient content in its parent. The footnote text is not an integral part of the paragraph; its existence is incidental to the determination of any actual semantic content in the element. Since without the footnote, the above paragraph is empty, it is considered to be empty by the Semantic Analyzer.

Conversely, in the first example, the "pubno" element is not considered empty, since both its sub-elements constitute semantic content for "pubno".

An element is considered empty if it has no #PCDATA content itself, and:

1. It has no sub-elements OR
2. It has one or more sub-elements, but all are considered to be insufficient content.

2.2 Element Content Pattern Matching

The Semantic Analyzer has at its disposal the full pattern-matching capabilities of the XGML Translator. These capabilities are used by the Semantic Analyzer to apply restrictions on the text that can occur in particular elements.

For instance, the following document excerpt contains text that does not conform to the intent of MIL-M-28001:

```
<SECTION ID="SCOPE1.">
<TITLE>1.  SCOPE</TITLE>
```

The numbering designation "1." clearly does not constitute a part of the section's title; it is expected that whatever application processes this SGML document will apply the numbering itself.

The Semantic Analyzer will detect titles starting with a variety of numbering patterns, and report the content as illegal.

2.3 Attribute Checking

There are two type of attribute processing provided by the Semantic Analyzer.

First, a particular attribute can be required to appear, in effect performing the task of the "REQUIRED" keyword in the DTD's attribute declaration.

Second, the pattern matching capabilities available for element content checking are also available for checking attribute values.

For instance, the "texttype" attribute from the body attribute set (common to many MIL-M-28001 elements) requires its content to consist of a two-digit code. There is no way to enforce this in the attribute declaration, but the Semantic Analyzer uses its pattern-matching abilities to insist on a two-digit format.

2.4 Processing Instruction Checking

Processing instructions can optionally be prohibited by the Semantic Analyzer, or pattern matching capabilities can be used to verify their contents.

For instance, if a class of documents is known to occasionally contain a processing instruction of the form:

```
<?Page 4-1-2>
```

the Semantic Analyzer can be instructed to report occurrences.

3. Semantic Analyzer Operation

This section provides an overview of the various stages of the analysis process, then details the features of the language used by the Semantic Analyzer.

3.1 Overview of the Analysis Process

The Semantic Analyzer is instructed to perform particular types of analysis, depending on the DTD being worked with, using an easily-learned specification language, described in subsequent sections.

A program known as the Semantic Analyzer Compiler processes the specification, and generates a corresponding XGML Translator program to do the actual analysis.

The generated XGML Translator program can then be used to analyze any documents in the class covered by the Analysis Specification.

3.2 Informal Language Syntax

The following sections present an informal overview of the Semantic Analyzer specification language. For an exact, formal description of the language, refer to Annex A.

All language keywords are case-independent.

3.2.1 Element Rule Overview

The analysis steps to be performed are broken down by elements from the DTD. The element rule is the fundamental structure in the specification language. Element rules describe the analysis to be performed on a particular element or group of elements.

An element rule for one particular element, "docno", for instance, starts like this:

```
element DOCNO
                    . . . element instructions . . .
```

If several elements receive the same analysis, they can be listed separated by commas, as follows (note the optional plural use "elements"):

```
elements STEP1, STEP2
                    . . . element instructions . . .
```

To analyze elements not specifically named in the element rules, the following is the proper form for the start of the element rule:

```
any other element
                    . . . element instructions . . .
```

The start of the element rule is followed by element classifications, element patterns, and attribute patterns, all described below:

3.2.2 Element Classifications

The element classification states the possible distinguishing features of an element. There are three possible classifications.

First, if the element is the document element, the element rule must contain the classification shown below:

```
element DOC
        is document element
                . . . more element instructions . . .
```

Second, if the element is declared to be "EMPTY" in the document's DTD, this must be stated in the element rule to prevent the Semantic Analyzer from reporting the empty content as an error.

```
element LEP
        is empty
                . . . more element instructions . . .
```

Third, as described in section 2.1 above, any elements considered to be insufficient content in their parent elements must be classified appropriately in the element rule:

```
element FTNOTE
        is note sufficient in its parent
                . . . more element instructions . . .
```

Note that it is allowed to classify an element as "empty" and as "insufficient"; all other combinations are invalid.

3.2.3 Element Patterns

Element patterns are used to match a given element's content against a stated pattern, and to allow or disallow the content based on the result of the comparison.

The pattern specification will state either that the element's content "must look like" the provided pattern, or that it "must not look like" it.

The patterns used in the specification must be enclosed in parentheses, and must be valid XGML Translator patterns; for a reference on pattern syntax and construction, refer to the XTRAN Programmer's Manual, chapter 11.

For example, in order to prevent occurrences of leading numbers at the start of a title, the following element rule could be used:

```
element TITLE
        is not sufficient in its parent
        must not look like (digit+ ["-" or "."] any-text*)
        . . . more element instructions . . .
```

3.2.4 Attribute Patterns

The simplest type of attribute analysis available is simply a requirement that the attribute be present. For example, even though the MIL-M-28001 DTD does not require the "security" attribute to be present for the "doc" element, an element rule can require it as follows:

```
element DOC
        is document element
        attribute SECURITY must be specified
        . . . more attribute patterns if desired . . .
```

The pattern specifications used for element content are also available for attribute values. For example, in order to require the "texttype" attribute's value to consist only of a two-digit code, the following attribute pattern would be used (this time, in the context of an element rule for "step1" and "step2" elements):

```
elements STEP1, STEP2
        attribute TEXTTYPE must look like (digit digit)
        . . . more attribute patterns if desired . . .
```

3.2.5 Processing Instruction Rule

A processing instruction rule begins as follows:

```
processing instructions
        . . . instructions . . .
```

In order to prohibit the appearance of processing instructions, the following processing instruction rule is used:

```
processing instructions
        not allowed
```

Otherwise, patterns may be specified for processing instructions, as done for elements above. To prohibit the appearance of processing instructions with pagination codes present, the following might be used:

```
processing instructions
        must not look like ("Page" white-space digit+ any-text*)
```

3.2.6 Comments

Comments may be included almost anywhere in a Semantic Analyzer specification, by enclosing the comment text on both sides with the character sequence "--". For example:

```
elements STEP1, STEP2
        -- allow only double-digit texttype codes --
        attribute TEXTTYPE must look like (digit digit)
        . . . more attribute patterns if desired . . .
```

In cases where the "--" sequence may be a valid part of a name, it will be treated as such and the comment text will then be interpreted as a specification instruction. For instance:

```
elements STEP1, STEP2-- Step Processing --
        attribute TEXTTYPE must look like (digit digit)
                . . . more attribute patterns if desired . . .
```

Since "STEP2--" is a valid SGML element name, the Semantic Analyzer compiler will not view the leading "--" as the start of a comment, and the text "Step Processing" will cause an error.

3.2.7 Macros

The Semantic Analyzer compiler provides a simple macro facility for use in Semantic Analyzer specifications.

A macro is useful when particular analysis instructions are recurring in a specification; for instance, the test for attribute "texttype" above is likely to recur in any element using the body attribute set. A macro for the test would be defined as follows:

```
common text-type
{
        attribute TEXTTYPE must look like (digit digit)
}
```

Macro names can be of arbitrary length, are case insensitive, and can consist of letters, digits, and the "-" and "." characters. The text enclosed between "{" and "}" delimiters is the macro expansion text.

To use the macro, its name is prefixed with an "@" symbol at any place in the specification where the expansion is valid. For example, the "text-type" macro could be used in the "step1" and "step2" element rule above as follows:

```
elements STEP1, STEP2-- Step Processing --
        @text-type
        . . . more attribute patterns if desired . . .
```

Macros must be defined in the specification before they are used.

APPENDIX A

A. Formal Language Syntax

The following productions specify the formal grammar of the Semantic Analyzer specification language. White-space is allowed to appear between any lexical tokens.

[1] analysis specification =
(element rule ¦ common definition ¦ processing instruction rule)+

[2] element rule =
named element rule ¦ implied element rule

[3] named element rule =
("ELEMENT" ¦ "ELEMENTS"), name list, element instructions

[4] implied element rule =
"ANY OTHER ELEMENT", element instructions

[5] element instruction =
element classification?, element pattern*, attribute pattern specification*

[6] element classification =
(document element classification ¦ parent content classification)?, empty classification?

[7] document element classification =
"IS DOCUMENT ELEMENT"

[8] parent content classification =
'IS NOT CONTENT IN ITS PARENT"

[9] empty classification =
'IS EMPTY"

[10] element pattern =
pattern specification

[11] attribute pattern specification =
"ATTRIBUTE" name, attribute requirement?, pattern specification*

[12] attribute requirement =
"MUST BE SPECIFIED", condition?

```
[13] pattern specification =
          ("MUST LOOK LIKE" ¦ "MUST NOT LOOK LIKE"), pattern

[14] pattern =
          pattern text, condition?

[15] condition =
          "WHEN", condition text

[16] pattern text =
          pattern component, ("OR" pattern component)*
          "(", character*, ")"

[17] condition text =
          "(", character*, ")"

[18] common definition =
          "COMMON", name, "(", character*, ")"

[19] processing instruction rule =
          "PROCESSING INSTRUCTIONS", ("NOTE ALLOWED") ¦ pattern
          specification+)

[20] common reference =
          "@", name

[21] comment =
          "--", data character*, "--"

[22] name list =
          name, (",",name)*

[23] name =
          As defined in ISO8879-1986

[24] data character =
          Any character

[25] pattern component =
          "(", data character*, ")"

[26] white space =
          (tab ¦ space ¦ record-end ¦ comment)+
```

APPENDIX B

B. Running the Semantic Analyzer

The Semantic Analyzer is provided on an MS-DOS floppy diskette, which also contains a number of batch files to facilitate the installation and analysis process.

To install the Semantic Analyzer, create the destination directory, CD to it, then type:

```
XCOPY B:\*.* /S
```

(Substitute "A:" if appropriate).

The file XSETUP.BAT should then be edited to reflect local path names.

The following batch files should then be run, in the specified order. They assume that XTRAN.EXE exists on the current path.

```
BLDSA

BLDCALS
```

The result is CALS.XNV, an XTRAN saved environment that allows Semantic Analysis of MIL-M-28001 documents. The file CALSDEMO.BAT provides an example of the analysis procedure.

See the individual batch files for further technical documentation.

APPENDIX C

C. Sample Specification

```
--    Semantic Analyzer Specification
      MIL-M-28001 CALS ANALYZER
      Version 1.0 11 APR 91

--

element DOC
      is document element

element TITLE
      is not sufficient in its parent
      -- Trap numbering at start --
      must not look like (white-space* digit+ [".-"]
                          any-text*)
      must not look like (white-space* uc "-" digit+
                          white-space+ any-text*)
      -- Trap table numbering --
      must not look like
           (white-space* ul "TABLE" white-space*
           digit+ [".-"] any-text*)
                when (parent is TABLE)
      -- Trap figure numbering --
      must not look like
           (white-space* ul "FIGURE" white-space*
           digit+ [".-"] any-text*)
                when (parent is FIGURE)

element FTNOTE
      is not content in its parent

element PARATEXT
      -- Trap step labels mis-coded in paratext --
      must not look like (white-space* lc "." any-text*)

-- Empty Elements --

element SEAL is empty
element LEP is empty
element CONTENTS is empty
element ILUSLIST is empty
element TABLIST is empty
element STDTABLE is empty
element COLHDDEF is empty
```

```
element COLBDDEF is empty
element STEMPH is empty
element ENDEMPH is empty
element STEMG is empty
element ENDEMG is empty
element FTNREF is empty
element XREF is empty
element GRAPHIC is empty
element BPWARN is empty
element BPCAUT is empty
element BPNOTE is empty
element INDEX is empty
element STCHG is empty
element ENDCHG is empty

any other element
     -- do nothing special --
```

APPENDIX D

1. INSTALLATION

To install the Semantic Analyzer, create the destination directory, CD to it, then type:

```
XCOPY B:\*.* /S
```

(Substitute "A:" if appropriate).

The file XSETUP.BAT should then be edited to reflect local path names.

The following batch files should then be run, in the specified order. They assume that XTRAN.EXE exists on the current path.

```
BLDSA
BLDCALS
```

The result is CALS.XNV, an XTRAN saved environment that allows Semantic Analysis of MIL-M-28001 documents. The file CALSDEMO.BAT provides an example of the analysis procedure.

See the individual batch files for further technical documentation.

2. DISK CONTENTS

read.me	This file
bldsa.bat	Builds Semantic Analyzer compiler
bldcals.bat	Builds Semantic Analyzer CALS application
xsetup.bat	Batch file called by batch files above
calsdemo.bat	Sample of analysis procedure
sa.dtd	DTD used internally by the Semantic Analyzer
sa.xt	Semantic Analyzer compiler
sa.xti	Include file for SA.XT
sau.xti	Include file for SA.XT
a50.sgm	The MIL-M-28001 DTD
entities.iso	XTRAN library file relating formal id's to pathnames
cals.sa	CALS Semantic Analysis specification
entities	Directory containing ISO entity declarations